Status of the new CRS software *(update)*

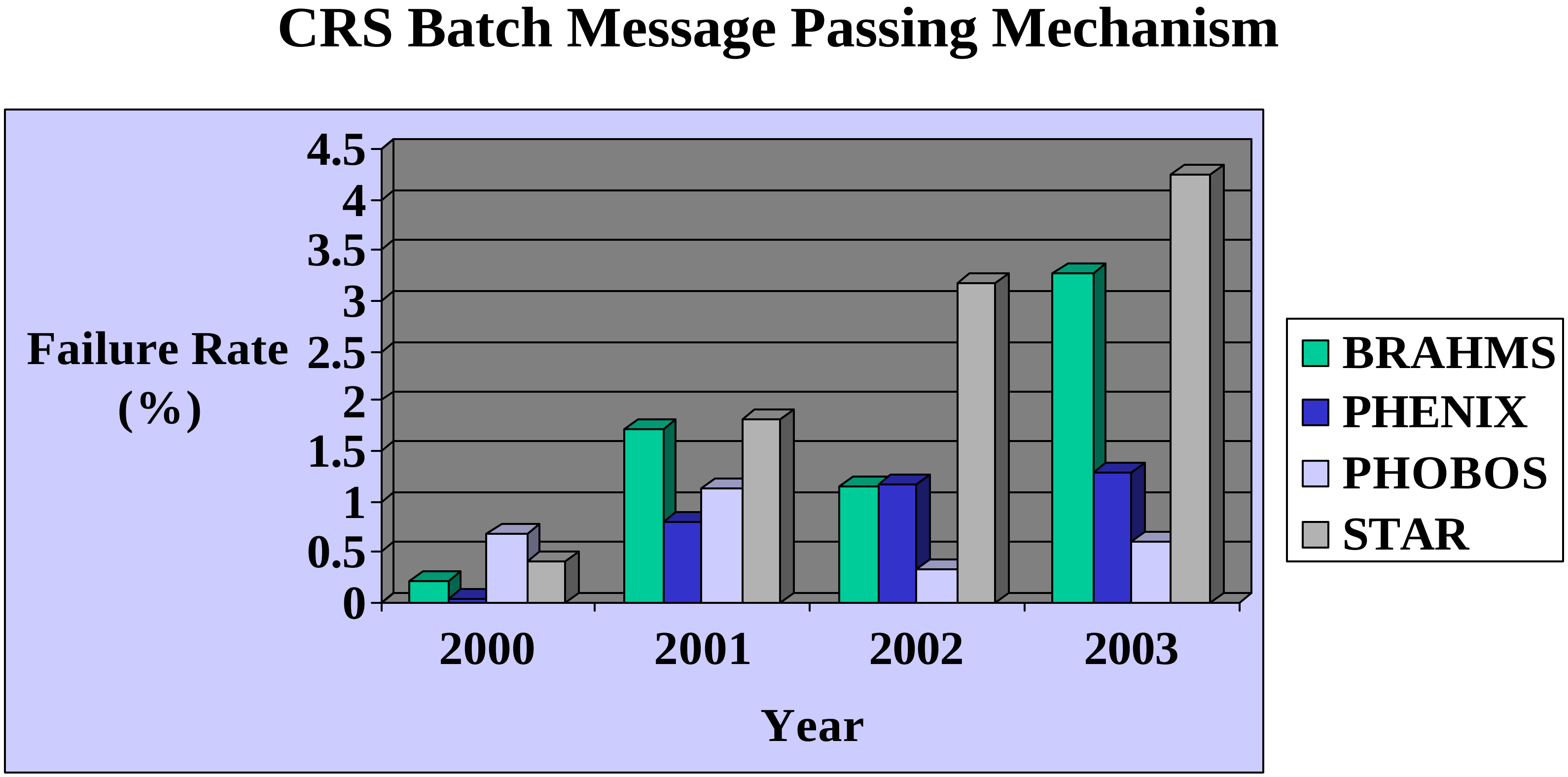
Tomasz Wlodek

June 22, 2003

Reminder

- Currently for data reprocessing we use a home grown batch system, called CRS
- We expect that this system will not scale well when we add new machines
- We would like to replace it by a new product
- PROBLEM: data needed for a job needs to be staged in from HPSS storage before the job execution starts
- Existing batch systems do not have this capability

Current CRS Batch Software does not scale with increasing farm size



New CRS will be based on Condor batch system.

Why?

1. Condor is free – can be used to replace LSF
2. Condor can be used both in CAS and CRS nodes, blurring distinction between those systems
3. Atlas…
4. <u>Condor comes with DAGMAN.</u>

What the hell is dagman?!

Condor on CAS machines

- **We need new LSF licenses**
- **They cost money**
- **We will try to use condor as the batch system on CAS machines. At present we are investigating this as an option, if we find out that Condor can be used in place of LSF we will do it.**

CAS/CRS

- At present the analysis machines run LSF while reconstruction machines run CRS
- If all machines run condor, then the distinction becomes blurred
- Depending on need we can easily send analysis jobs to CRS machines or CRS to CAS
- Farm usage will become more efficient.
- *Ofer Rind will work on Condor for CAS*

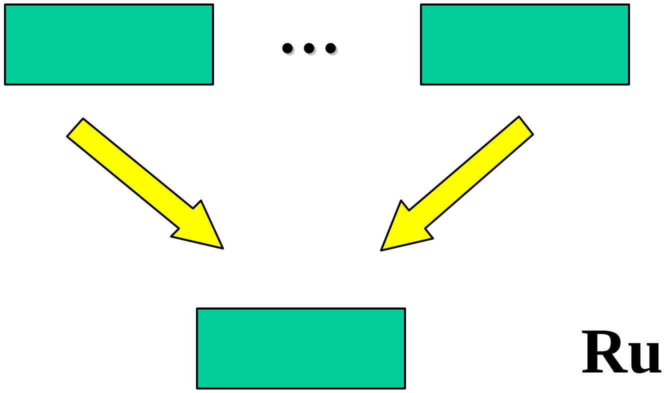
Atlas work…

- Condor is de facto standard in the Grid community
- It may become standard in the future Atlas work.
- It is worthwhile to have some expertise with it
- Deploying a big (few k nodes) Condor farm may be a first step towards building a grid-enabled farm. *(At least this is my dream)*

The most important reason why use Condor in CRS:

- Condor comes with Dagman – an utility for chaining complex “graphs of jobs
- The reason why CRS software was necessary was the need to “prestage” data from HPSS, before running reconstruction jobs.
- Dagman solves this problem for us.

How does a reconstruction job look like?



…

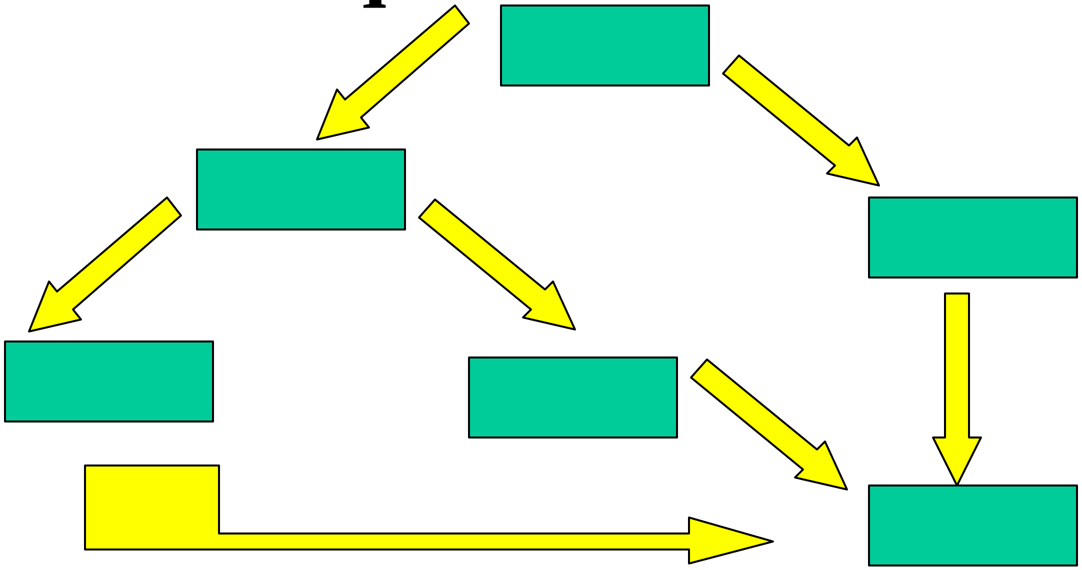
Get data (From HPSS, disk,…)

Run the job

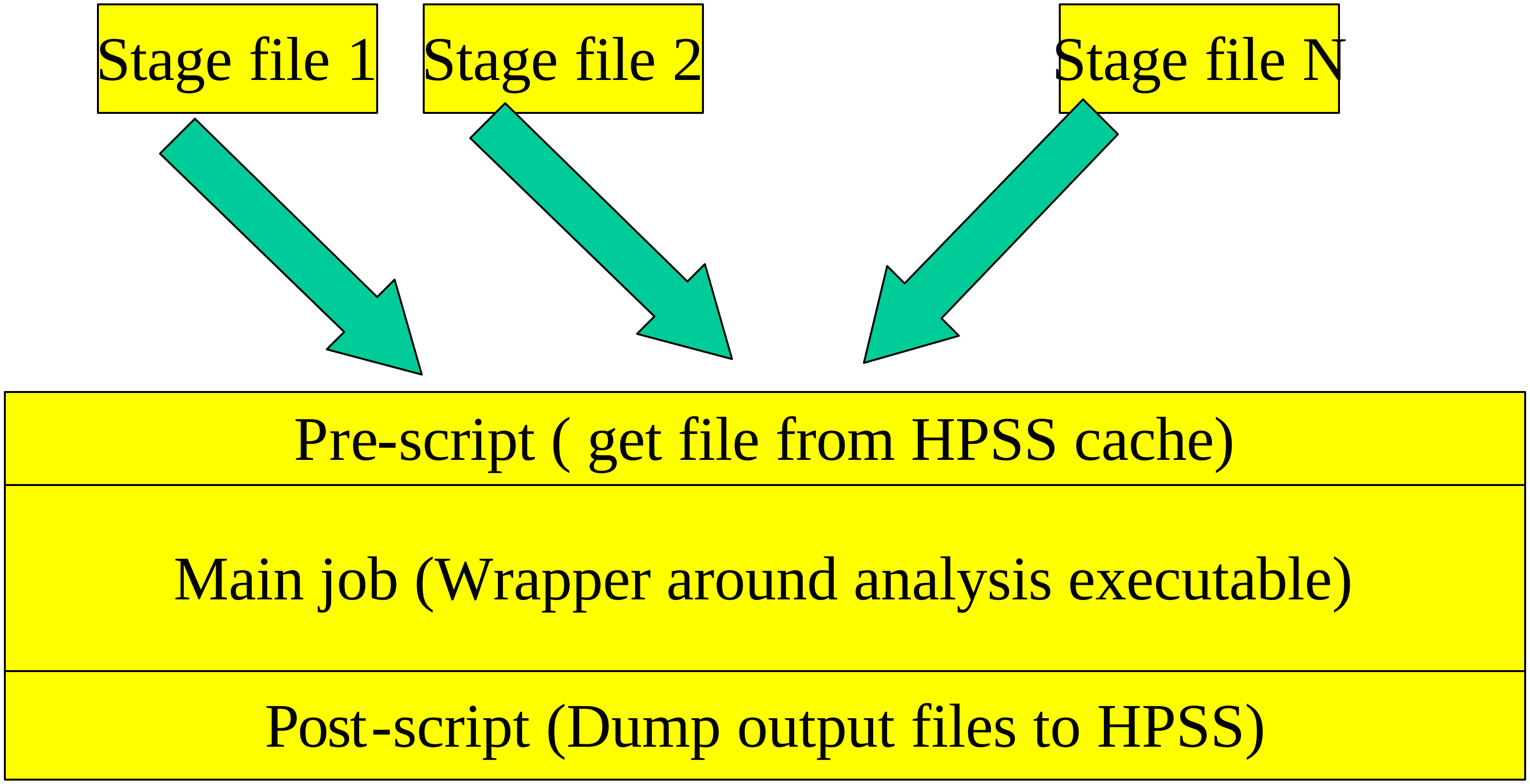
Main job cannot run until all data is available
This is our main problem.
Solution comes with Dagman.

Condor+Dagman

- Dagman is a tool you add on top of Condor batch system which allows you to build graphs of interconnected jobs which will execute in sequence



How a CRS job will look in DAGMAN



Stage file 1

Stage file 2

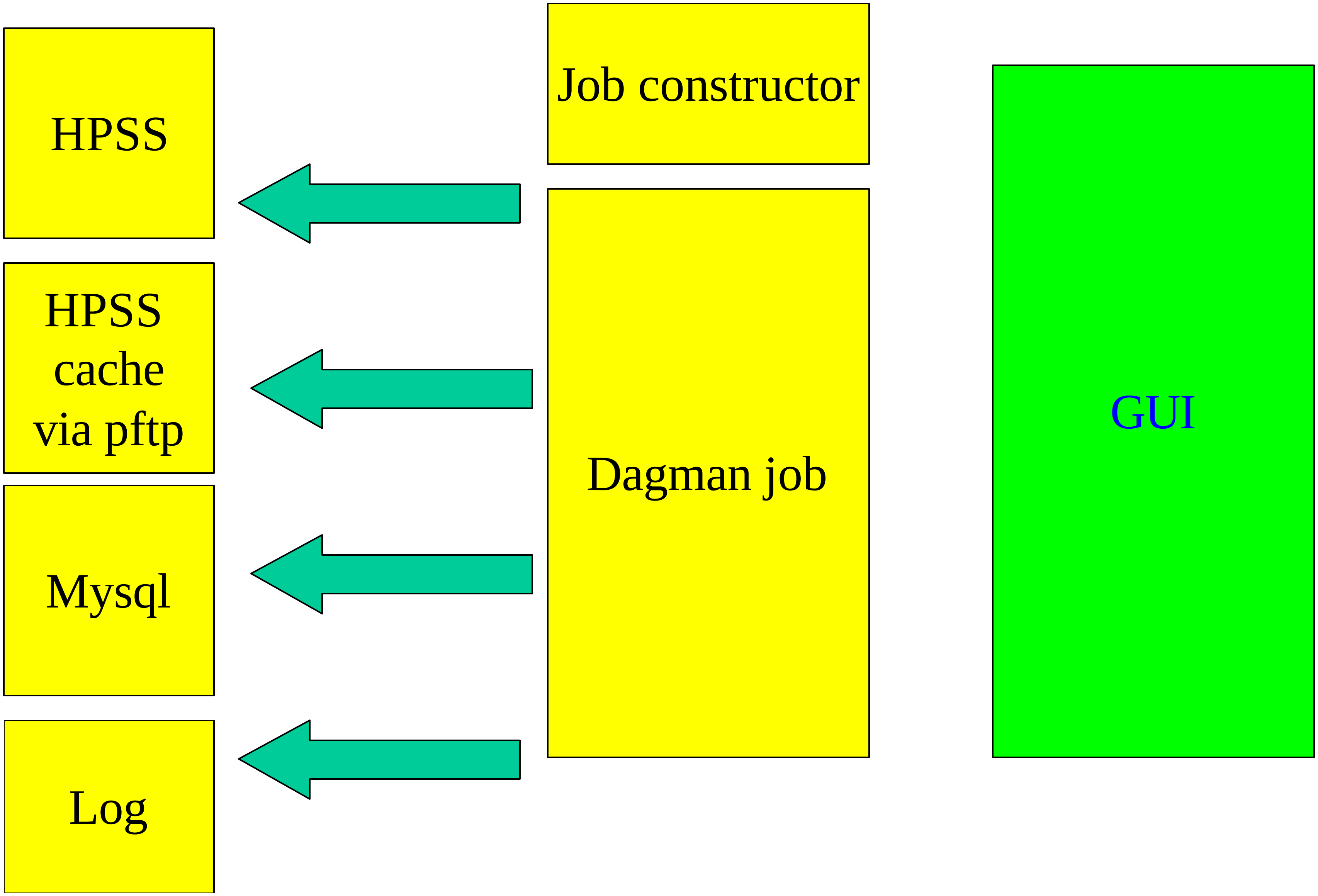
Stage file N

Pre-script (get file from HPSS cache)

Main job (Wrapper around analysis executable)

Post-script (Dump output files to HPSS)

What other elements we need?



CRS components

- Job constructor
- Interface to HPSS
- Interface to HPSS cache/PFTP wrapper
- Interface to MYSQL
- Log file
- GUI for job submission and control

Job constructor

- Takes a standard job definition file (same as in old CRS) and writes a set of python scripts for importing input files and a script for the main job.
- Writes a set of job definition files for condor.
- Create a job directory, submit condor/dagman job, mark the status of jobs in MySQL job database
- Status: Done.

```
#!/usr/bin/env python

import os, stat, sys, shutil, string, commands, time

sys.path.append("/direct/u0b/tomw/crs")

from ssh_link import *

from logbook  import *

from mysql    import *

# Details about creation info

$CREATION_INFO

# routine for setting up the environment

$SETENV_ROUTINE

def get_input_file_from_hpss(InputFile,LocalFileName):

        HpssLogbook=LOGBOOK(os.environ["CRS_LOGBOOK"],"pftp","INFO“)
        Database=MYSQLDB(os.environ["CRS_EXPERIMENT"])

        JobName=os.environ["CRS_JOB_NAME"]
```

```
#!/usr/bin/env python
import os, stat, sys, shutil, string, commands, time
sys.path.append("/direct/u0b/tomw/crs")
from ssh_link import *
from logbook  import *
from mysql    import *
# Details about creation info
#Created from :1056373909_star.jdl
#Date of creation:Mon Jun 23 09:11:50 2003
#Machine:rplay11.rcf.bnl.gov
# routine for setting up the environment
def set_env():
# routine which sets up environment variables for CRS job
  os.environ["CRS_JOB_FILE"]="star.jdl"
  os.environ["OUTPUT3"]="st_physics_4070053_raw_0010001.tags.root"
  os.environ["CRS_PFTP_SLEEP"]="1.0"
  os.environ["CRS_PFTP_SERVER"]="hpss"
  os.environ["CRS_EXECUTABLE"]="/u0b/tgtreco/crs/bin/run_reco.csh"
  os.environ["CRS_MYSQL_PWD"]="_node_update"
  os.environ["CRS_MYSQL_USER"]="star_reco_node"
```

HPSS interface

- Request a file from HPSS, check if the request was accepted, wait for response, decode the response and if successful – return to main DAG
- If the file requested is not a HPSS file but a disk located file – check if the file exist and is available
- Status: Done.

MySQL interface

- **Store information about jobs being executed**
- **We will use MySQL version 4.0.12, more modern than the one used with old CRS**
- **Not all databases used by old CRS are needed anymore**
- **The job database from old CRS needs to be slightly modified**

New MySQL databases

- Old CRS job database has three tables: Jobs,subjobs and files.
- Each table will receive new entry: Dag-name, corresponding to the name of the entire dag built from single condor jobs.
- Status: Written, however not yet fully integrated with the rest of the code. (Will be done in matter of days).

Logfiles

- For debugging purposes each dag will maintain a human readable logfile of all its activities. The logfiles will contain information about status of various components of the dag.
- Status: done.

Pftp wrapper to access HPSS cache

- Contact HPSS cache, check if requested file exist, get it to local disk, check if the transfer was successful, retry if necessary, return error code if not
- Same thing when transferring data into HPSS
- Status: Done.

GUI

- GUI will be used to follow status of a job, see status of the farm and so on.
- GUI will obtain status information from CRS-jobs MySQL database.
- I plan to re-use the GUI from old CRS, however I am not yet sure how many modifications are needed to make it work.
- If to many changes are needed I will write a new one myself.
- Status: Either already exist or needs to be written from scratch. (Will know soon)

Integration of various components

- I have made attempts to run real analysis jobs from Star and Brahms
- No success – the jobs depend on many initializations
- I decided to finish the code first and only after the coding is fully done return to running experiments jobs.

Integration - cd

- **So far I can run jobs using Tom Throwe’s test executable (reads binary input file, spits some output, runs in a loop for a while)**
- **The test executable works fine, gets inputs from HPSS and UNIX**
- **Some flaws in our Condor installation discovered and fixed**

I can now go back and run experiments jobs.

- I plan to start running experiment jobs now.
- Hopefully after debugging I can start small scale production
- For mass production I need to have the GUI ready.

Next steps:

- I will add new input/output file type: GRIDFILE, allowing jobs to grab I/O from any place on the net
- But this is low priority now, first get new CRS running.

Current status of Condor:

- We use Condor 6.4.7, latest stable version.
- CRS development is done on 3-node condor installation
- We have Condor installed on 192 nodes, running under single master.
- We are learning now how to define queues for different experiments

Condor Status - cd

- The coming version of Condor (6.5.3) will have “condor on demand” feature, allowing for near interactive work
- User submits a request, which briefly takes over the nodes, executes, and returns control to batch system
- Atlas people expressed interest in that …
- Towards end of year we can upgrade Condor, so if Rhic people are interested in Computing on Demand – you can have it.

Conclusions:

- **New CRS slowly takes shape**
- **Hopefully I will start data production soon – but I will need help (and patience!) from experiments.**
- **We will gradually increase the size of CRS production test bed, so that by end of year all data production is taken over by new CRS.**
- ***Condor for CAS – subject to another talk.***